Department: Case Studies in Translational Computer Science
Editor: Manish Parashar, manish.parashar@utah.edu; David Abramson, david.abramson@uq.edu.au

PyCOMPSs as an instrument for Translational Computer Science

Rosa M Badia
Barcelona Supercomputing Center

Javier Conejero
Barcelona Supercomputing Center

Jorge Ejarque
Barcelona Supercomputing Center

Daniele Lezzi
Barcelona Supercomputing Center

Francesc Lordan
Barcelona Supercomputing Center

***Abstract*—With the advent of distributed computing, the need for frameworks that facilitate its programming and management has also appeared. These tools have typically been used to support the research on application areas that require them. This poses good initial conditions for Translational Computer Science (TCS), although this does not always occur. This paper describes our experience with the PyCOMPSs project, a programming model for distributed computing. While it is a research instrument for our team, it has also been applied in multiple real use cases under the umbrella of European Funded projects or as part of internal projects between various departments at the Barcelona Supercomputing Center (BSC). The paper illustrates how the authors have engaged in TCS as an underlying research methodology, collecting experiences from three European projects.**

PROGRAMMING parallel and distributed computing systems is a challenging task. Many aspects contribute to it: the complexity of the computing infrastructure with new architectures and heterogeneous devices, the compute and data distribution aspects, or the complexity of the applications which need to leverage the underlying infrastructure power.

To address these challenges, multiple groups have been conducting research towards providing programming environments that simplify the de-

http://dx.doi.org/10.1109/MCSE.2022.3152945

velopment of applications [1], [2].

Among the paradigms to ease the development of parallel applications, a widely-supported approach is task-based programming. Based on defining parallelism at the task level, a task may have different granularities: from a few lines of code to a function, to an invocation to an external binary. Most environments consider identifying data dependencies between tasks and build a Directed Acyclic Graph (DAG) at execution, with nodes representing tasks and edges data-dependencies between them. The paradigm has proven to be applicable both at node level [3], [4] and on distributed computing environments [5], [6].

The PyCOMPSs project[1] was started 15 years ago. One of the project's goals is to produce stable and reliable software that end-user applications can use. This gives us feedback driving new research and developments in the project while at the same time enabling progress in the application research areas.

This paper describes our research methodology in programming environments for distributed computing and how translational research in computer science (TCS) has guided our research process [7].

Overview

PyCOMPSs [8], [9] is a programming environment designed and developed at the Barcelona Supercomputing Center (BSC). Based on the tasks paradigm, the primary goal of this software is to ease the development of parallel applications for distributed computing platforms.

Depending on the granularity of the tasks, the environment can be used to develop traditional task-based applications (fine-grain tasks) or to develop workflows (coarse grain tasks).

With the PyCOMPSs project, we have conducted research on multiple topics, among others: the programming model itself; resource management and its execution in High-Performance Computing (HPC), clouds, containers; task scheduling; integration with storage and Input/Output (I/O); or convergence of HPC, big data and artificial intelligence.

[1]http://compss.bsc.es/,, PyCOMPSs is the Python binding of COMPSs. For clarity, in this paper, PyCOMPSs is used as a generic term which includes COMPSs

While the project does not target a specific application area, we have been working in the context of multiple projects with user communities that have provided specific requirements. In this sense, PyCOMPSs have been leveraged in project use cases in biomedicine, engineering, biodiversity, chemistry, astrophysics, financial, telecommunications, manufacturing, and Earth sciences.

Although PyCOMPSs is an academic open-source research project, the code is managed under a Continuous Integration Continuous Deployment (CI/CD) process with a testing infrastructure that validates new features before merging. Periodic stable releases are delivered twice per year. We perform multiple training activities, and support is provided under a best-effort approach by the team members.

BSC is proud of having projects that live beyond their funding schemes. For example, BSC's performance tools have been developed for more than 25 years and PyCOMPSs for around 15 years now. While specific developments from basic research projects may be only done as prototype versions, most of those from funded projects are integrated into the official versions.

Translation process

TCS in European (EU) funded projects

From its early times when COMPSs was started in the CoreGRID[2] project, the framework has been involved in more than 25 EU-funded projects. While Abramson and Parashar [7] identified general shortcomings in existing funding schemes for supporting TCS, some EU funding schemes support projects that include translational computing in some sense.

The European Commission (EC) not only funds basic research programs, such as the prestigious ERC or the Marie Curie. For example, the current H2020 program considers two types of schemes: Research and Innovation Actions (RIA) and Innovation Actions (IA).

RIAs fund more research-oriented activities but still expect industry involvement to explore the possible industrial feasibility of the research results. In IAs, funding focuses on closer-to-the-market activities, including prototyping, testing, demonstrating, piloting, etc.

[2]http://coregrid.ercim.eu/

In addition, the "HPC Centres of Excellence" (CoE) address scientific applications and user communities running application codes or large scale workloads seeking extreme scaling performance.

These are three examples of the funding schemes in Europe. All of them are collaborative in nature, expecting a consortium that consists of universities, research institutes, Small and Medium Enterprises (SME), and large companies. The distinct partners can play different roles in the consortium: research or technology provider, application provider, end-user, etc. In many cases, the industry plays the end-user role, providing use cases that are somehow prototyped and evaluated in the project. This does not mandate TCS, but from the authors' point of view, it may help in the process.

The EC indicates in the call text the expected impact of the funded projects, for example, indicating the level of innovation and productivity enhancement that is sought.

In this article, we will focus on three exemplar EU funding experiences on which we have been involved with PyCOMPSs: The BioExcel CoE, the ExaQUte RIA project and the eFlows4HPC IA project (see figure 1).

The BioExcel CoE

BioExcel[3] is the HPC European Centre of Excellence for Computational Biomolecular Research. BioExcel's mission is to provide applications, tools, support, and networking opportunities to address grand scientific challenges that fully exploit the power of large e-infrastructures.

Our team focuses on a collaboration with the Institute for Research in BioMedicine (IRB) towards the design and development of the BioExcel Building Blocks (BioBB) library [10].

Each BioBB is a Python wrapper on top of biomolecular simulation tools. The building blocks share a unique syntax, requiring input files, output files and input parameters, irrespective of the wrapped program. Workflows assembled by composing multiple building blocks and packaged in a single Python script with a defined Conda environment to be shared and reproduced. Multiple workflow engines can enact BioBB workflows, specifically PyCOMPSs, when targeting HPC environments.

During the development of the BioBB library, the BioBB development team and the PyCOMPSs team have been working in collaboration. While the BioBB development team has conducted its research on workflows for biomolecular research, the PyCOMPSs team has performed more generic research on the resource management and execution of large workflows in HPC systems.

Multiple specific research topics in the PyCOMPSs team have arisen thanks to the BioBB workflow's requirements. Some of these topics were related to fault tolerance: management of application failures at task-level, support to application restart, and application checkpointing. For example, the original behaviour of PyCOMPSs applications was to safely terminate the whole application if a task error was detected. The BioBB developers indicated to the PyCOMPSs team that such behavior was too conservative and would like to enable the workflows to continue the execution even when some tasks failed. We extended the PyCOMPSs syntax to enable the developer to indicate the desired behaviour if an error occurred in a task [11]. For example, this interface allows you to tell the runtime to ignore individual tasks' errors and continue execution and cancel the execution of erroneous task successors.

The BioBB library is offered to the user community through tutorials, as ready-to-use workflows, or as source code to build new workflows. An example of the level of readiness of the BioBB workflows has been demonstrated with the SARS-CoV-2 emergency. In early 2020 the two teams worked together to define a set of pre-exascale workflows. However, the pandemic changed the priorities towards research questions related to the virus's evolutionary path or the different human sensitivity reactions. The BioBB and their workflows were quickly adapted to answer these questions, and they were run in the MareNostrum 4 supercomputer at BSC.

ExaQUte project

The ExaQUte project aimed at constructing a framework to enable uncertainty quantification and optimization in complex engineering problems, using computational simulations on exas-

[3] www.bioexcel.eu

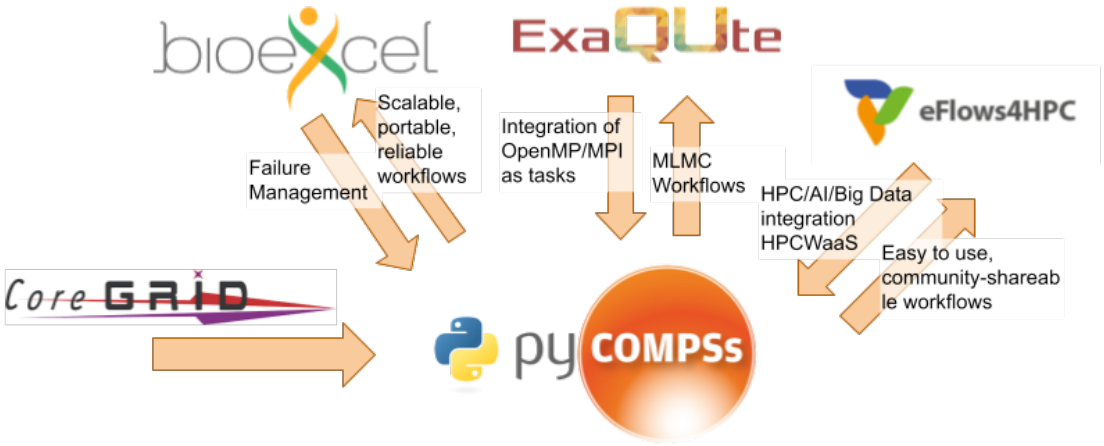

Figure 1: PyCOMPSs and its relation to exemplar EU projects

cale systems. The project ran from May 2018 to the end of 2021.

The project was based on Multi-Level Monte Carlo (MLMC) to enable a large number of stochastic variables. The MLMC algorithm is implemented with the xMC library[4], which explores multiple simulations of the Kratos multiphysics software[5]. Kratos is fed with meshes of different characteristics defined with the ParMmg software[6]. The whole framework is integrated with Python scripts annotated with PyCOMPSs decorators to support parallelism and distributed computing.

Concerning PyCOMPSs, the main goal was the extension of existing task schedulers to extract the parallelism of the MLMC algorithm and to support distinct levels of complexity of the tasks (OpenMP and MPI tasks), which implies different duration and different amounts of resources used by each task. An critical paradigm to support was the relaxation of global synchronizations required between simulations belonging to different levels of the MLMC algorithm [12]. In addition, the support for MPI tasks was very naive at that time and required extensions to support more complex data layouts.

The research in the project was conducted at multiple levels, with sub-objectives, mapping to the components mentioned before (xMC, Kratos, ParMmg and PyCOMPSs). While the components were providing requirements to others, the main driver was a final user application aimed at robust optimization of structures subject to wind action.

Although ExaQUte involved a small number of partners, it had the typical consortium structure, with an end-user from industry, research and technology providers, and two supercomputer centers providing infrastructure. The final user was the SME `str.ucture`, whose main engineering activities lie around the industrial application of advanced computer-aided simulation methods on parallel HPC platforms. The company has leveraged the research performed in the project to study different use cases of its interest, like the assessment of wind-induced galloping instability of cable cars or the wind effects on large span bridges. This second research was done in collaboration with German structural engineering companies.

eFlows4HPC project

With the experience obtained in previous projects, our research group proposed the eFlows4HPC project. The technical objective of the project goes beyond PyCOMPSs and proposes a whole software stack for the development of workflows that involve HPC simulation and modeling, artificial intelligence, and big data analytics.

The project aims to demonstrate through three application Pillars with high industrial and social relevance how the realization of forthcoming efficient HPC and data-centric applications can be

[4] https://doi.org/10.5281/zenodo.3235832
[5] www.cimne.com/kratos/
[6] www.mmgtools.org/

developed adopting new workflow technologies.

The project will integrate existing workflow interfaces, programming models, artificial intelligence, and data analytics libraries to provide a uniform, easy-to-use platform that enables the exploitation of future large-scale systems.

The project also contributes with the HPC Workflows as a Service (HPCWaaS) idea to widen newcomers' access to HPC, and in general, to simplify the deployment and execution of complex workflows in HPC systems, providing mechanisms to enable the sharing, reuse, and reproducibility of complex workflows.

The project involves three application pillars stakeholders to define complex workflows based on the project technologies. While validating the technologies, these workflows are also a vehicle for research on the pillar topics: digital twins for manufacturing, climate prediction, and urgent computing for natural hazards.

Another goal is the user communities' adoption of the project solutions defined to enable impact on industrial cases and their exploitation in future HPC systems. To reinforce the feedback to communities, the CoEs ChEESE[7], ESiWACE[8] and EXCELLERAT[9] are involved in the project, in coordination with the Focus CoE[10]. To this end, the project activities include organizing workshops and training schools that will transfer the project methodologies and results to the relevant CoEs and industrial communities, contributing to the reduction of skill gaps in Europe related to HPC and workflows development.

Impact

The paper has described several cases where PyCOMPSs is used in translational computer science research. For the activities in the BioExcel project, the impact of the translational process is evident for us. All the stakeholders involved in the activities have benefited from it. PyCOMPSs has been enhanced with new features, that were fresh and exciting enough to imply innovative research contributions [11]. A significant plus for our team is that these new features were helpful for the BioBB workflows and have fostered the research and development of workflows for molecular dynamics. Finally, these activities impact the user community, with new workflows available for their research, and have been helpful for emerging research activities, such as the COVID investigations.

For the ExaQUte project, similar conclusions can be derived. The project put together a whole software stack integrating components from different partners. Beyond the research on programming models for distributed computing, other partners conducted research on multi-level Monte Carlo or simulation of structures, to name a few. Multiple requirement-feedback loops were exchanged between the whole consortium partners. A small spin-off project, EdgeTwinsHPC[11], was funded to explore the viability of HPC software to generate digital twins that run on the edge. Part of this research is also conducted in Pillar I of the eFlows4HPC project.

We hope that in a couple of years we will also be able to describe similar success stories for the eFlows4HPC project which, from our point of view, has been designed driven by translational computing methodology.

Lessons learnt

Our experience shows that the research translation is an iterative process in which new ideas appear after some work. The implementation of the new ideas enables further developments that can provide fresh ideas, and so on. This turns out to have benefits to the different teams involved in the translation process.

One of the challenges of translational research is the multidisciplinary aspect that requires some adjustments in the way of working. The progress sometimes seems to be slow due to different communication languages, priorities, or skills (i.e., what is easy for one group looks very difficult for another). One lesson learned in this sense is that patience is essential. One should not get disappointed if the initial results do not look as expected.

Another lesson that we have learned is that at BSC we are privileged because we have multiple research departments working on different topics in the same place. While working with

[7] www.cheese-coe.eu
[8] www.esiwace.eu
[9] www.excellerat.eu
[10] https://www.hpccoe.eu/about/
[11] www.edgetwins.eu

cross-discipline teams requires some effort, the impact and results exceed the investment. In this sense, our recommendation would be to foster the foundation of inter-disciplinary centers' and ask their groups to work together in joint projects.

We recognize a trade-off between the cost of delivering stable software, usable for others, and the number of research publications. The effort is not always recognized, at least in the short term. Nevertheless, it is worth it: besides offering exciting tools to the community, starting a new research topic from stable, reliable software is a powerful beginning. The group has been able to deal with this and at the same time enjoy enough flexibility and creativity to conduct research on new topics.

Conclusion

This paper has presented the authors' view of the Translational Computer Science methodology with the PyCOMPSs project. PyCOMPSs is a parallel programming model for distributed computing developed at BSC. Since its infancy, it has been partially supported by multiple EU-funded projects. Instead of proposing a brand new independent idea for each project, which is developed as a prototype discontinued after the funding period, the approach has been to allow the project to continue alive thanks to multiple cycles of funding.

In addition, we have leveraged the privileged situation at BSC with multidisciplinary research departments and the collaborative nature of most EU funding schemes to implement a TCS approach.

Acknowledgement

This work has been supported by the Spanish Government (contract TIN2015-65316-P), by the Generalitat de Catalunya (contract 2014-SGR-1051), and by the European Commission's Horizon 2020 Framework program through BioExcel Center of Excellence (contracts 823830, and 675728), the ExaQUte project (contract 800898) and the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955558 and by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR.

REFERENCES

1. E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, et al., Pegasus: A framework for mapping complex scientific workflows onto distributed systems, Scientific Programming 13 (3) (2005) 219–237. doi:10.1155/2005/128026.
2. Y. Babuji, A. Woodard, Z. Li, D. S. Katz, B. Clifford, R. Kumar, L. Lacinski, R. Chard, J. M. Wozniak, I. Foster, et al., Parsl: Pervasive parallel programming in python, in: Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing, 2019, pp. 25–36. doi:10.1145/3307681.3325400.
3. A. Duran, E. Ayguadé, R. M. Badia, J. Labarta, L. Martinell, X. Martorell, J. Planas, Ompss: a proposal for programming heterogeneous multi-core architectures, Parallel processing letters 21 (02) (2011) 173–193. doi:10.1142/S0129626411000151.
4. P. Bellens, J. M. Perez, R. M. Badia, J. Labarta, Cellss: a programming model for the cell be architecture, in: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, 2006, pp. 86–96. doi:10.1145/1188455.1188546.
5. R. M. Badia, J. Labarta, R. Sirvent, J. M. Pérez, J. M. Cela, R. Grima, Programming grid applications with grid superscalar, Journal of Grid Computing 1 (2) (2003) 151–170. doi:10.1023/B:GRID.0000024072.93701.f3.
6. E. Tejedor, R. M. Badia, Comp superscalar: Bringing grid superscalar and gcm together, in: 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID), IEEE, 2008, pp. 185–193. doi:10.1109/CCGRID.2008.104.
7. D. Abramson, M. Parashar, Translational research in computer science, Computer 52 (9) (2019) 16–23. doi:10.1109/MC.2019.2925650.
8. F. Lordan, R. M. Badia, et al., ServiceSs: an interoperable programming framework for the Cloud, Journal of Grid Computing 12 (1) (2014) 67–91. doi:10.1007/s10723-013-9272-5.
9. E. Tejedor, R. M. Badia, J. Labarta, et al., PyCOMPSs: Parallel computational workflows in Python, The International Journal of High Performance Computing Applications 31 (2017) 66–82. doi:10.1177/1094342015594678.
10. P. Andrio, A. Hospital, J. Conejero, L. Jordá, M. Del Pino, L. Codo, S. Soiland-Reyes, C. Goble, D. Lezzi, R. M. Badia, et al., Bioexcel building blocks,

a software library for interoperable biomolecular simulation workflows, Scientific data 6 (1) (2019) 1–8. doi:10.1038/s41597-019-0177-4.

11. J. Ejarque, M. Bertran, J. Á. Cid-Fuentes, J. Conejero, R. M. Badia, Managing failures in task-based parallel workflows in distributed computing environments, in: European Conference on Parallel Processing, Springer, 2020, pp. 411–425. doi:10.1007/978-3-030-57675-2_26.
12. R. Tosi, R. Amela, R. M. Badia, R. Rossi, A parallel dynamic asynchronous framework for uncertainty quantification by hierarchical monte carlo algorithms, Journal of Scientific Computing 89 (1) (2021) 1–25. doi:10.1007/s10915-021-01598-6.